

# **Intel® Itanium® Processor Abstraction Layer**

**Tony Luck**  
**Principal Engineer**  
**April 18<sup>th</sup>, 2007**



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications. Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The **Itanium architecture processors** may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's website at <http://www.intel.com>.

**Intel and Itanium** are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Copyright © 2007, Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.



# Agenda

- **Overview of Itanium® Architecture  
PAL Firmware**
- **PAL calls**
  - What they do
  - How Linux uses them
- **Itanium Architecture Firmware Flows**
  - Reset sequence
  - Machine check abort

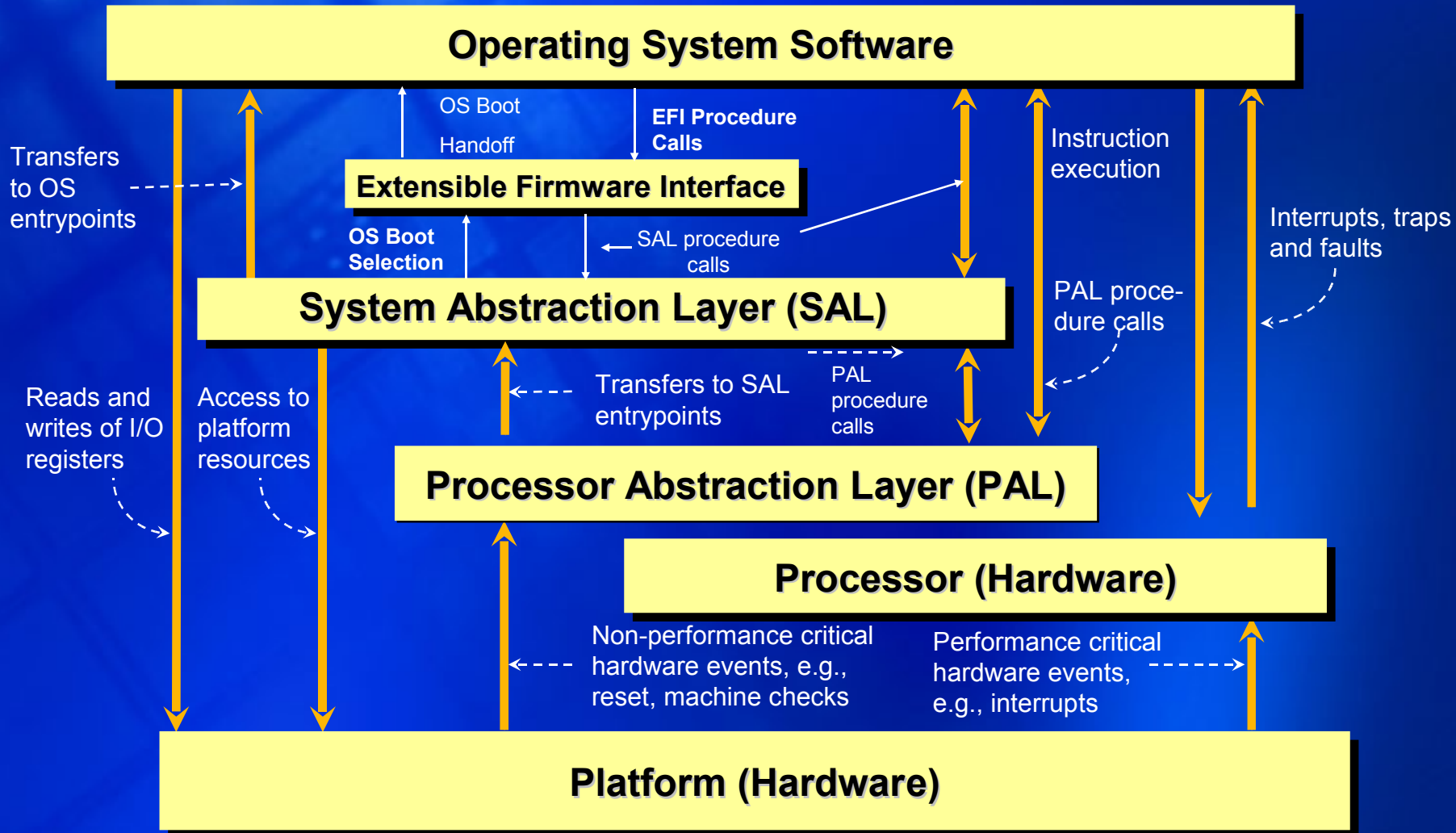


# Why is Itanium® Architecture-Based Firmware Different?

- **IA-32 firmware**
  - Developed in an ad-hoc manner
  - No explicit IA-32 firmware architecture
  - Loosely defined interfaces
- **Itanium® architecture-based firmware**
  - Explicit architecture
  - Abstracts changes in processor and platform implementations



# Itanium® Architecture-Based Firmware



# Processor Abstraction Layer

- PAL provides a consistent interface to processor features
  - PAL interface is defined in the *Intel® Itanium® Architecture Software Developer's Manual*





# PAL Functionality

- **Implements non-performance critical processor functions**
  - Simplifies hardware implementation
  - Allows silicon to be used for performance enhancements
- **PAL functions**
  - Processor configuration
  - Processor feature identification
  - Processor self-test and initialization
  - PAL machine check error handling
  - Processor power management
- **Intel provides PAL binary**
  - PAL authentication prevents PAL modifications
  - PAL version checking requires a stepping-appropriate PAL release



# PAL Components

- **PAL\_A**

- Entry points for Reset, Init, and machine check
- Minimal PAL procedures for recovery
- Stepping independent
  - Errata workarounds that may be provided in PAL\_A
  - PAL\_A is revised during development
- May be placed in protected boot block

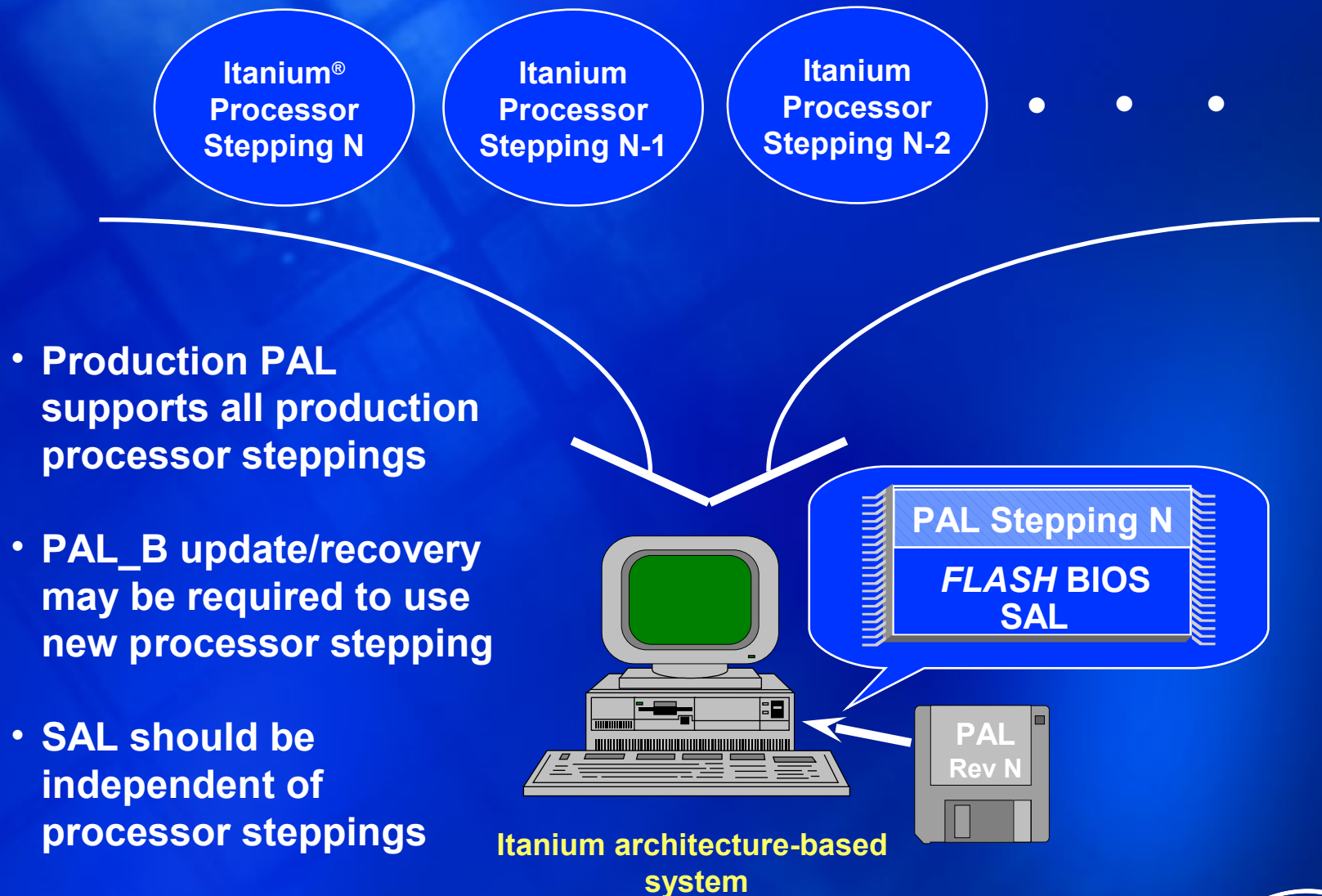
- **PAL\_B**

- Entry point for PMI
- Reset, Init, PMI, and machine check handlers
- Remaining PAL procedures
- Stepping dependent
- Production PAL supports all previous production steppings





# PAL and SAL vs. Processor Steppings



# PAL Entrypoints

- **Reset**
  - Power-on and warm reset
- **Init**
  - Typically used for “crash dump”
  - If supported by OS, Init flow is continuable
- **Machine Check Abort**
  - Machine checks requiring PAL, SAL, or OS intervention
- **PMI**
  - Replaces IA-32 System Management Mode (SMM)
- **Virtualization**
  - See PAL\_VM\_\* calls



# PAL Procedures

- Provides APIs to processor-specific functionality
- Calling convention
  - All PAL calls may be made in physical mode
  - Some PAL calls may be called in virtual mode
  - SDM indicates whether PAL procedure is called using static or stacked registers
  - PAL calls may be made cacheable or uncacheable
- On following slides mark those used by Linux († = used, ‡ = used only in /proc)



# PAL Info and Config Procedures

## *PAL PROCEDURE*

- **PAL\_BUS\_GET\_FEATURES ‡**
- **PAL\_BUS\_SET\_FEATURES**
- **PAL\_DEBUG\_INFO ‡**
- **PAL\_FIXED\_ADDR**
- **PAL\_FREQ\_BASE ‡**
- **PAL\_FREQ\_RATIO†**

## *FUNCTIONALITY*

Return configurable processor bus interface features and their current settings

Enable or disable configurable features in processor bus interface

Return the number of instruction and data breakpoint registers

Return the fixed component of a processor's directed address

Return base frequency of platform if generated by processor

Return the ratio of processor, bus and interval time counter to base platform frequency



# PAL Info and Config Procedures (Continued)

## *PAL PROCEDURE*

- **PAL\_PERF\_MON\_INFO†**
- **PAL\_PLATFORM\_ADDR**
- **PAL\_PROC\_GET\_FEATURES†**
- **PAL\_PROC\_SET\_FEATURES**
- **PAL\_REGISTER\_INFO ‡**
- **PAL\_RSE\_INFO†**
- **PAL\_VERSION ‡**

## *FUNCTIONALITY*

- Return the number of and type of performance monitors
- Specify processor interrupt block address and I/O port space address
- Return configurable processor features and their current settings
- Enable or disable configurable processor features
- Return AR and CR register information
- Return RSE information
- Return the MINIMUM PAL version needed by the processor





# PAL Info and Config Procedures (Continued)

## *PAL PROCEDURE*

## *FUNCTIONALITY*

- **PAL\_LOGICAL\_TO\_PHYSICAL†** Returns information on the logical to physical processor mapping
- **PAL\_CACHE\_SHARED\_INFO†** Returns information on caches shared between logical processors





# PAL Machine Check Procedures

## *PAL PROCEDURE*

- **PAL\_MC\_CLEAR\_LOG**
- **PAL\_MC\_DRAIN†**
- **PAL\_MC\_EXPECTED**
- **PAL\_MC\_DYNAMIC\_STATE**
- **PAL\_MC\_ERROR\_INFO**
- **PAL\_MC\_RESUME**
- **PAL\_MC\_REGISTER\_MEM**

## *FUNCTIONALITY*

Clear all error info from processor error logging registers

Ensure that all operations that could cause a MCA have been completed

Set/Reset expected machine check indicator

Return processor dynamic state for logging by SAL

Return processor machine check info and processor static state for logging by SAL

Restore minimal architected state and return to interrupted process

Register minimal state save area with PAL for machine checks and inits



# PAL Cache and Memory Procedures

## *PAL PROCEDURE*

- **PAL\_CACHE\_FLUSH†**
- **PAL\_CACHE\_INFO†**
- **PAL\_CACHE\_INIT**
- **PAL\_CACHE\_PROT\_INFO**
- **PAL\_CACHE\_SUMMARY†**
- **PAL\_MEM\_ATTRIB ‡**
- **PAL\_PTCE\_INFO†**
- **PAL\_VM\_INFO ‡**

## *FUNCTIONALITY*

- Flush instruction or data caches**
- Return detailed instruction or data cache information**
- Initialize the instruction or data caches**
- Return instruction or data cache protection information**
- Return a summary of the cache hierarchy**
- Return a list of supported memory attributes**
- Return info needed for PTCE instruction to purge entire TC**
- Return detailed virtual memory features supported in the processor**



# PAL Cache and Memory Procedures

(Continued)

## *PAL PROCEDURE*

- **PAL\_VM\_PAGE\_SIZE†**
- **PAL\_VM\_SUMMARY†**
- **PAL\_VM\_TR\_READ ‡**

## *FUNCTIONALITY*

Return virtual memory TC and hardware walker page sizes supported in the processor

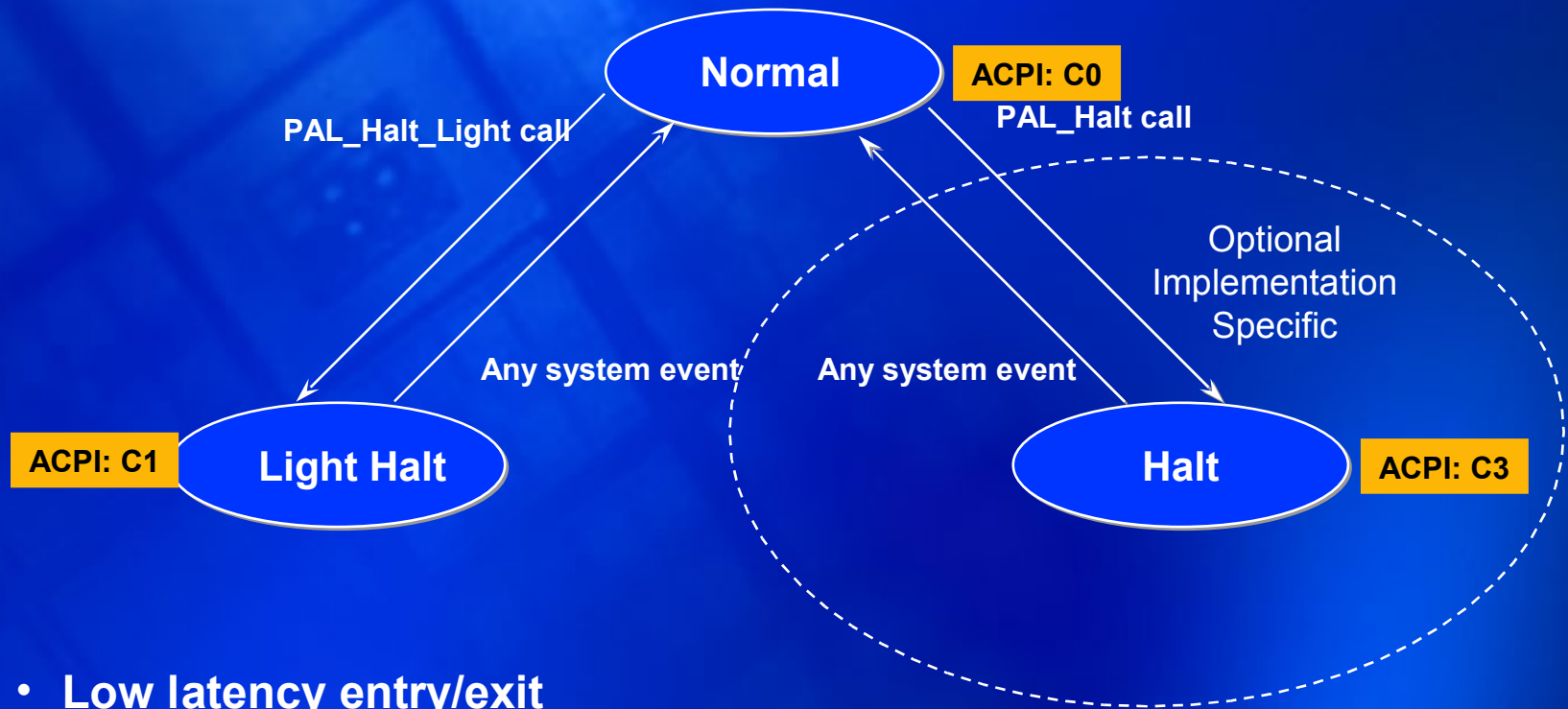
Return summary of virtual memory features supported in the processor

Read contents of a translation register



# Itanium<sup>®</sup> Architecture

## Power (C-state) Management



- Low latency entry/exit
- Instruction execution stopped
- Cache coherency maintained

# PAL Power Procedures

## *PAL PROCEDURE*

- **PAL\_HALT†**
- **PAL\_HALT\_LIGHT†**
- **PAL\_HALT\_INFO†**

## *FUNCTIONALITY*

**Enter the low power HALT state or an implementation dependent low power state**

**Enter low power Light HALT state**

**Return the low power capability of the processor**



# PAL Diagnostic Procedures

## *PAL PROCEDURE*

- **PAL\_CACHE\_LINE\_INIT**
- **PAL\_CACHE\_READ**
- **PAL\_CACHE\_WRITE**
- **PAL\_TEST\_INFO**
- **PAL\_TEST\_PROC**

## *FUNCTIONALITY*

**Initializes tags and data of a cache line for processor testing**

**Read tag and data of a cache line for diagnostic testing**

**Write tag and data of a cache for diagnostic testing**

**Return the amount of memory needed for late processor self-test**

**Perform late processor self-test**





# PAL Support Procedures

## *PAL PROCEDURE*

- **PAL\_COPY\_INFO**
- **PAL\_COPY\_PAL**
- **PAL\_PMI\_ENTRYPOINT**

## *FUNCTIONALITY*

**Return info needed to relocate PAL procedures and PAL PMI code to memory**

**Relocate PAL procedures and PAL PMI code to memory**

**Register PMI memory entrypoints with processor**



# More PAL calls

## *PAL PROCEDURE*

- **PAL\_PREFETCH\_VISIBILITY†**

## *FUNCTIONALITY*

- **Change virtual memory attributes**

# Newer PAL calls (in SDM 2.2)

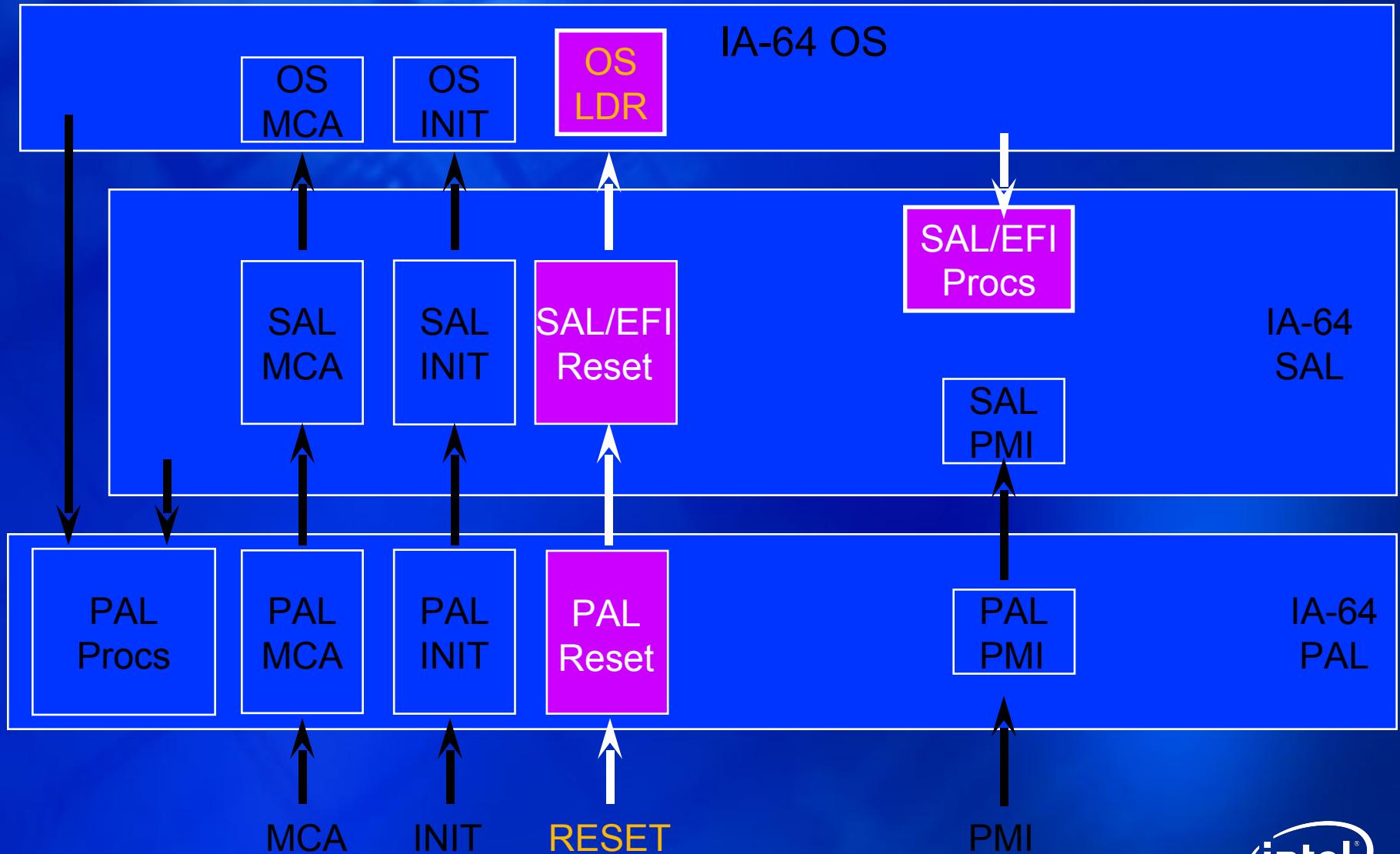
- PAL\_BRAND\_INFO ‡
- PAL\_GET\_HW\_POLICY
- PAL\_SET\_HW\_POLICY
- PAL\_GET\_PSTATE†
- PAL\_SET\_PSTATE†
- PAL\_PSTATE\_INFO
- PAL\_MC\_ERROR\_INJECT†
- PAL\_MEMORY\_BUFFER
- PAL\_SHUTDOWN
- Processor branding information
- H/w thread priority tuning
- Power (P-state) tuning
- Inject errors for testing
- Provide PAL a cacheable memory buffer
- Put processor into low power state until RESET



# PAL calls for Virtualization

- PAL\_VP\_CREATE
- PAL\_VP\_ENV\_INFO
- PAL\_VP\_EXIT\_ENV
- PAL\_VP\_INIT\_ENV
- PAL\_VP\_REGISTER
- PAL\_VP\_RESTORE
- PAL\_VP\_SAVE
- PAL\_VP\_TERMINATE
- PAL\_VPS\_\*
- Regular PAL calls to create, manipulate and terminate virtual processors
- Virtualization entry points into VMM from PAL

# Itanium® Architecture Reset Sequence



# Reset Sequence

Processor BIST



PAL\_A  
authentication



PAL\_B authentication



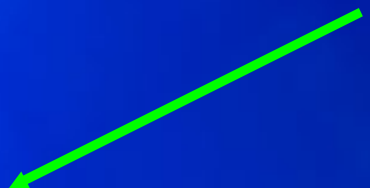
SAL recovery check /  
PAL\_B version check



PAL early self-test



SAL initializes memory



PAL late self-test



SAL continues boot



# Machine Check Abort Sequence

